510.84
Iℓ6r
no.1042-
1052
1980
Incompl.
c.3

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAY 0 4 1993		

L161—O-1096

510.84
Il6r
no. 1052
cop. 3

UIUCDCS-R-81-1052

UILU-ENG 81 1721

A Characterization of Globally Consistent Databases and their Correct Access Paths

by

Yehoshua Sagiv

July 1981

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Digitized by the Internet Archive
in 2013

http://archive.org/details/characterization1052sagi

UIUCDCS-R-81-1052

A Characterization of Globally Consistent Databases and their Correct Access Paths

Yehoshua Sagiv

Department of Computer Science

University of Illinois at Urbana-Champaign

Urbana, Illinois 61801

July 1981

This work was supported in part by NSF grant MCS-80-03308.

ABSTRACT

We propose the representative instance as a representation of the data stored in a database whose relations are not the projections of a universal instance. We characterize the database schemes for which local consistency implies global consistency. (Local consistency means that each relation satisfies its own functional dependencies, and global consistency means that the representative instance satisfies all the functional dependencies.) We show how to compute efficiently projections of the representative instance provided that local consistency implies global consistency. Throughout this work we assume that a cover of the functional dependencies is embodied in the database scheme in the form of keys.

1.0 Introduction

The universal instance assumption is essential to many papers in design theory for relational databases. As pointed out in [FMU], two different concepts are included in this assumption. The most basic concept is the universal relation scheme assumption (also known as the "uniqueness assumption" [Ber]). It asserts that each attribute has a unique role, that is, for any subset of attributes X, there is (at most) one relationship among the attributes of X. This assumption is made explicitly or implicitly in many papers in design theory for relational databases. In particular, it is made in papers dealing with the axiomatization of dependencies, and synthesis and decomposition of relation schemes.

The second and more controversial concept is the universal instance assumption, that is, the assumption that the relations of a database are the projections of a single relation over the set of all the attributes. This assumption is needed in order to define lossless joins [ABU]. There are two versions of this assumption. According to the first, this assumption has to be made only in order to determine whether a join is lossless, i.e., it is only a tool that is used when a database is designed and when queries are evaluated [FMU]. The second version (the pure universal instance assumption) states that the relations of a database must always be the projections of a universal instance, and null values have to be used in order to satisfy this requirement [HLY,Ko,Li,Ma,Sc].

When a universal instance is assumed, users usually formulate queries having in mind the universal instance rather than the actual relations of the database [KU]. If a given query refers to a set of attributes X, then the first step in evaluating this query is to compute the projection of the universal instance onto X. If the pure universal instance assumption is made, it is sufficient to take any lossless join over a set of attributes that contains X. If the relations of the database are not the projections of a universal instance, different lossless joins might give different results. In [FMU] this problem is solved by requiring that the join dependency consisting of all the relation schemes be acyclic. In this paper we propose an alternative solution. We believe that our solution reflects the properties of functional dependencies better than [FMU]. In particular, some of the problems left open in [FMU] are handled in our case without explicitly defining maximal objects [MU].

In this paper we assume a universal relation scheme, but not a pure universal instance. Instead we define the <u>representative instance</u> of a database (in Section 3). The representative instance has been used to determine whether the database satisfies a set of functional dependencies [Hol,Val]. We believe that the representative instance correctly describes the information stored in the database even when the relations are not the projections of a universal instance. As a first step toward evaluating queries with respect to the representative instance, we address the following problems.

(1) Under what conditions the database is globally consistent if each

relation is locally consistent? That is, under what conditions the representative instance satisfies all the functional dependencies if each relation satisfies its own functional dependencies?

(2) How can we compute efficiently projections of the representative instance?

Throughout this paper we assume that a cover of the functional dependencies is embodied in the database scheme in the form of keys (as in [Ber]). In Section 3 we define the uniqueness condition, and in Section 4 we prove that local consistency implies global consistency if and only if the database scheme satisfies the uniqueness condition. In Section 5 we show how to compute efficiently projections of te representative instance provided that the database scheme satisfies the uniqueness condition.

2. Preliminaries

2.1 Basic Definitions

We view a relation (cf. [C1]) as a finite table with columns labeled by attributes, and rows, called tuples, that represent mappings from the attributes to their associated domains. Let μ be a tuple of a relation labeled by a set of attributes R. If $A \in R$, then $\mu(A)$ is the value of μ in column A; if $S \subseteq R$, then $\mu[S]$ denotes the values of μ for the attributes in S. We say that r is a relation over a set of attributes R if the columns of r are labeled by the attributes of R.

We use the letters A,B,C,... to denote attributes, and the letters ...,X,Y,Z to denote sets of attributes. A string of attributes (e.g., ABCD) denotes the set containing these attributes, and the union of two sets X and Y is written XY.

A relational database scheme over a set of attributes U is a set of ordered pairs $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ such that $R_i \subseteq U$ and K_i is a set of (explicit) keys of R_i. (If $X \in K_i$, then $X \subseteq R_i$; and no key of R_i properly contains another key of R_i.) Each R_i is a set of attributes labeling the columns of a relation, and we use it as the name of the relation (i.e., there are no distinct relations over the same set of attributes). We call each R_i a relation scheme. The set of FDs (functional dependencies) of R_i is

$$F_i = \{X \rightarrow A \mid X \in K_i \text{ and } A \in R_i - X\}$$

A database is a set of relations $r_1,\ldots,r_n$ over $R_1,\ldots,R_n$, respectively, such that each r_i satisfies F_i. That is, a database is the "current value" of the database scheme. We assume that $F = \bigcup_{i=1}^{n} F_i$ is a cover of all the FDs imposed on the database by the user. In other words, a cover of the FDs is embodied in the database scheme in the form of (explicit) keys (as in [Ber]).

2.2 Relations with Null Values

In many cases there is a need to represent partial information in the database. If we have a relation over the attributes Manager and

Department, and Jones is a manager without a department, then the tuple (Jones,δ) is inserted into this relation. The value δ is a special value, called a <u>null value</u>, and it denotes unknown information. Suppose that there are two managers without a department, e.g., Jones and Smith. There is no reason to assume that they manage the same (unknown) department. In order to distinguish the null value in the tuple (Jones,δ) from the null value in the tuple (Smith,δ), we will mark each null value with a unique subscript and store the tuples (Jones,δ_1) and (Smith,δ_2). Null values with distinguishing subscripts are called <u>marked null</u> [Ko,Ma], and will be used exclusively in this paper. Two null values are equal only if they have the same subscript. We say that tuples μ_1 and μ_2 <u>agree</u> on column A, written $\mu_1(A) = \mu_2(A)$, if either both $\mu_1(A)$ and $\mu_2(A)$ are not null and equal or both are null and equal.

2.3 The Chase Process

Suppose that a relation r (with null values) is required to satisfy an FD $X \to A$. It is not necessarily correct to argue that r violates $X \to A$ if it has two tuples that agree on X and disagree on A. Instead, we should use $X \to A$ to equate symbols(2) of r in the following way. Suppose that r has tuples μ_1 and μ_2 that agree on all the columns for X but disagree on the column for A. If μ_1 has δ_i in column A and μ_2 has δ_j in column A, then we can replace all occurrences of δ_j in r with δ_i. If μ_1 has a non-null value c in column A and μ_2 has a null value δ_j in

(2) Occasionally, we refer to null and non-null values as symbols.

that column, then we can replace all occurrences of δ_j with the non-null value c.

Suppose that the relation r is required to satisfy a set F of FDs. We can apply the FDs of F to r until no more symbols of r can be equated. The relation obtained in this way is called the chase of r with respect to F, written $\text{chase}_F(r)$, and it satisfies an FD $X \rightarrow A$ of F if and only if there is no pair of tuples that agree on X and disagree on A. We say that the relation r satisfies F if and only if $\text{chase}_F(r)$ satisfies F. If r satisfies F, then $\text{chase}_F(r)$ is unique up to renaming of null values [MMS].

Example 1: Suppose that $F = \{A \rightarrow C, A \rightarrow D, B \rightarrow C, CD \rightarrow B\}$ and let r be the following relation.

A	B	C	D
1	1	δ_1	δ_2
δ_3	2	1	δ_4
1	δ_5	δ_6	2
2	1	δ_7	1

We apply the FD $A \rightarrow C$ to the first and third tuples to replace δ_6 with δ_1. Then all occurrences of δ_1 are replaced with δ_7 by applying $B \rightarrow C$ to the fourth and first tuples. The result is the following relation.

A	B	C	D
1	1	δ_7	δ_2
δ_3	2	1	δ_4
1	δ_5	δ_7	2
2	1	δ_7	1

By applying $A \rightarrow D$ to the first and third tuples, δ_2 is replaced with 2. Now δ_5 is replaced with 1 by applying $CD \rightarrow B$ to the first and third tuples. Since no more applications are possible, $chase_F(r)$ is

A	B	C	D
1	1	δ_7	2
δ_3	2	1	δ_4
1	1	δ_7	2
2	1	δ_7	1

Clearly, $chase_F(r)$ (and hence r) satisfies F. If we add the FD $C \rightarrow D$ to F, then $chase_F(r)$ is no longer unique, since δ_2 can be replaced with either 1 or 2, and $C \rightarrow D$ is not satisfied by $chase_F(r)$ (and r). []

If r satisfies F, then r also satisfies additional FDs that can be inferred by Armstrong's axioms [Arm]. The <u>closure</u> of a set of attributes X with respect to a set of FDs F, written X_F^+, is the set of all attributes A such that $X \rightarrow A$ can be derived from F by Armstrong's axioms. We can compute X_F^+ in linear time [BB]. If F denotes a cover of all the FDs imposed on the database (i.e., the embodied FDs), then we usually write X^+ instead of X_F^+.

2.4 Relational Expressions and Extension Joins

In this paper we consider relational expressions over the operators project, (natural) join, and union that are denoted by π, $\bowtie$, and $\cup$, respectively. The operands are the relation schemes $R_1,\ldots,R_n$. Let α be a set of relations $r_1,\ldots,r_n$ for the relation schemes $R_1,\ldots,R_n$, respectively, such that each r_i satisfies F_i. The value of E for α, written $v_\alpha(E)$, is computed by substituting the relations $r_1,\ldots,r_n$ for the relation schemes $R_1,\ldots,R_n$, and applying the operators according to the usual definitions. (When the join is applied, two tuples are joined on a given column only if they agree in this column.) Two expressions E_1 and E_2 are(1) equivalent, written $E_1 \equiv E_2$, if for all sets α, $v_\alpha(E_1) = v_\alpha(E_2)$. The expression E_2 is contained in E_1, written $E_2 \subseteq E_1$, if for all sets α, $v_\alpha(E_2) \subseteq v_\alpha(E_1)$. An expression E has a unique value for the current database and, so, by a slight abuse of notation we denote this value by E (rather than $v_\alpha(E)$, where α is the set of the current relations). In particular, if μ is a tuple in the value of E for the current database, then we write $\mu \in E$.

The expression $\bowtie_{j=1}^{m} R_{i_j}$ is an extension join [Ho2] of R_{i_1} if for all $1 \leq j < m$, the set $R_{i_1} \ldots R_{i_j}$ contains a key of $R_{i_{j+1}}$. Extension joins are a special case of lossless joins [ABU].

Example 2: Let <ABC,{A}>,<BD,{B}>,<CDE,{CD}> be a database scheme. ABC $\bowtie$ BD is an extension join of ABC. Similarly, ABC $\bowtie$ BD $\bowtie$ CDE is an extension join of ABC, since ABC contains the key B of BD, and ABCD contains the key CD of CDE. ABC $\bowtie$ CDE is not an extension join, since ABC does not contain any key of CDE (i.e., ABC does not contain CD which is the only key of CDE). []

(1) This notion of equivalence is called strong equivalence in [ASU].

3. The Representative Instance

The ultimate goal of designing a database scheme has always been a collection of relation schemes $R_1, \ldots, R_n$ that are independent, that is, relation schemes that allow the user to update each relation in the database without having to change the contents of the other relations. Of course, there might be some semantically meaningful constraints (e.g., as in [C2]) that do not allow every possible update. But these constraints should be as limited as possible. We feel that enforcing the universal instance assumption is too restrictive. Clearly, this assumption can always be enforced by using marked nulls [KU,Ma]. However, this can be done only at the expense of applying the chase process to the universal instance whenever updates are performed on the database. Furthermore, there is a need to store many null values that do not provide any information. These null values are needed only to satisfy the universal instance assumption.

Efficiency is not the only issue. Most relational database systems are not designed to use the chase process. Therefore, there is a need to develop a theory for determining correct access paths when the universal instance assumption is not satisfied. The simplified universal instance assumption of [FMU] is one possible solution. In [FMU] a certain condition is imposed on the database scheme, and it is claimed that under that condition a minimal (in the number of relation schemes) lossless join is a correct access path (even if the relations of the database are not the projections of a universal instance). We will present a different set of conditions and show that queries should be evaluated by performing the union of several lossless joins rather than just one lossless join.

In this section we will define the representative instance [Ho1,Va2] of a database $r_1,\ldots,r_n$. The representative instance is defined for every $r_1,\ldots,r_n$ (even if the r_i´s are not the projections of a single relation). If $r_1,\ldots,r_n$ are the projections of a universal instance r and the database scheme has the lossless join property, then r is also the representative instance. In [Ho1,Va1] the representative instance is used to determine whether the database satisfies a set of FDs. We believe that the representative instance is also a correct representation of all the information stored in the database, and queries posed about the contents of the database should be answered with respect to the representative instance.

Let U be the set of all the attributes. A relation r_i can be viewed as a relation over U by adding columns for the attributes in $U - R_i$ that contain distinct null values. Formally, the augmentation of a relation r_i over R_i to a relation over U, written $\alpha_U(r_i)$, is

$$\{\mu \mid \mu \text{ agrees with some tuple of } r_i \text{ on } R_i, \text{ and has distinct null values (that do not appear in any other tuple) for the attributes of } U - R_i\}$$

Example 3: Let r be the relation

A	C
c_1	c_2
c_3	c_4

$\alpha_{ABCD}(r)$ is the relation

A	B	C	D
c_1	δ_1	c_2	δ_2
c_3	δ_3	c_4	δ_4

where $\delta_i \notin \{c_1,c_2,c_3,c_4\}$ for every i. []

Consider a database $r_1,\ldots,r_n$ over relation schemes $R_1,\ldots,R_n$, and let $r' = \bigcup_{i=1}^{n} \alpha_U(r_i)$. If there are no dependencies, then r′ is the <u>representative instance</u> of the database $r_1,\ldots,r_n$. When dependencies are present, the chase process should be applied to r′. Thus, if the only dependencies are those in the set of FDs F, then the representative instance is $\text{chase}_F(r')$. The database $r_1,\ldots,r_n$ satisfies the set of FDs F if the representative instance satisfies F [Hol,Val].

<u>Example 4</u>: Consider the database scheme <ABCD,{A}>,<CGDEF,{CG}>,<DEFB,{DEF}>,<BCF,{BC}>. Note that this database scheme is in Boyce-Codd normal form. Suppose that the relation for ABCD is {1112}, the relation for CGDEF is {11111}, the relation for DEFB is {1111}, and the relation for BCF is empty. To obtain the representative instance we have to compute the chase of the following relation.

A	B	C	D	E	F	G
1	1	1	2	δ_1	δ_2	δ_3
δ_4	δ_5	1	1	1	1	1
δ_6	1	δ_7	1	1	1	δ_8

The null value δ_5 can be replaced with 1 by applying the FD DEF → B to the second and third tuples, and then δ_2 is replaced with 1 by applying BC → F to the first and second tuples. No FD can be applied after that, and so the representative instance is

A	B	C	D	E	F	G
1	1	1	2	δ_1	1	δ_3
δ_4	1	1	1	1	1	1
δ_6	1	δ_7	1	1	1	δ_8

This representative instance (and hence also the database) satisfies the FDs that are embodied in the relation schemes. However, if the relation for BCF were {112} instead of ϕ, then the representative instance would violate the FD BC → F. []

In the above example, the projection of the representative instance onto BCF contains the tuple 111. It may be argued that the tuple 111 over the attributes BCF does not represent a correct information, since the relation for the relation scheme BCF does not contain this tuple. However, if this argument is accepted, then it follows that two distinct relationships between the attributes B, C, and F are stored in the database. One relationship is stored in the relation for the relation scheme BCF, and the other relationship is obtained by the extension join CGDEF $\bowtie$ DEFB. But this is contrary to the universal relation scheme assumption that is essential to many works in design theory for relational databases. Without this assumption the axioms for functional and multivalued dependencies, and the various synthesis and decomposition algorithms cannot be used. Therefore, we believe that the representative instance correctly represents the information stored in the database.

3.1 Computing Total Projections of the Representative Instance

In the remainder of this paper we consider a database scheme $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ and a corresponding database $r_1,..,r_n$ with a representative instance r. For simplicity´s sake, we assume that the relations $r_1,..,r_n$ do not contain null values. (Thus, null values exist only in the representative instance.) We can always decompose a relation scheme into smaller relation schemes, each having the keys of the original relation scheme and some of its attributes. Therefore, the only actual restriction implied by our assumption is that null values cannot be stored for the attributes of a key.

The user formulates queries having in mind the representative instance r rather then the individual relations $r_1,\ldots,r_n$. Suppose that the user is formulating a query that refers to a set of attributes X. The first step in evaluating this query is to compute the projection of r onto X. We assume that if the user is referring to the attributes in X, then he/she is interested only in tuples of r that have non-null values for all the attributes in X. Therefore, the problem addressed in this paper is how to compute the X-total projection of r, i.e.,

$$\{\mu \mid \mu \text{ is a tuple in } \pi_X(r) \text{ without any null value}\}$$

For example, given the database of Example 4, the ACF-total projection of the representative instance is {111}.

Example 4 illustrates two surprising facts. First, the representative instance does not necessarily satisfies the functional dependencies even if each relation satisfies the FDs imposed by its keys (this fact was originally pointed out in [Hol]). Second, an X-total projection of the representative instance cannot always be computed by joining several relations of the database and then projecting onto X. In that example, the ABCDF-total projection of the representative instance is {11121}. However, no expression of the form $\pi_X(\bowtie_{j=1}^{m} R_{i_j})$ has as its value the relation {11121} over ABCDF. (Here, $R_{i_1},\ldots,R_{i_m}$ are some of the relation schemes.) In the following sections we will characterize those cases in which the problems indicated by Example 4 do not occur.

3.2 The Uniqueness Condition

Consider a tuple μ of $\bigcup_{j=1}^{n} \alpha_U(r_j)$ that has originated from the relation r_i. That is, μ is non-null in the columns of R_i and has distinct nulls in all the other columns. Therefore, a null value in column $A \in U - R_i$ of μ can be replaced during the chase process (either with another null or with a non-null) only if $A \in R_i^+$ [BDB]. The problems illustrated in Example 4 are caused by the existence of two different ways that could potentially replace a specific null value. In order to avoid such troublesome situations, we require that for each relation scheme R_i, there is a unique way to derive R_i^+. We will show that the representative instance of the database $r_1,\ldots,r_n$ satisfies F (i.e., the set of all the embodied FDs) if and only if each R_i^+ is uniquely derived.

We will define "uniqueness" after the following informal example.

Example 5: Consider the algorithm of Fig. 1. This algorithm computes X^+ assuming that a cover of F is embodied in the relation schemes. Suppose the database scheme is <ABC,{A}>,<BD,{B}>,<CD,{C}>. The closure of ABC can be computed (i.e., derived) in two different ways. Either we use BD and its key B to obtain ABCD from ABC in line 3 of Fig. 1, or we use CD and its key C to obtain ABCD from ABC.

If the database scheme is <ABC,{A}>,<BD,{B}>,<CE,{C}>, then there is only one way to compute $(ABC)^+$. BD is used to add D, and CE is used to add E. []

We say that R_i satisfies the uniqueness condition if there is no R_j ($j \neq i$) such that for some set of attributes X and an attribute A

(1) $XA \subseteq (R_i)^+_{F-F_j}$, and

(2) $X \in K_j$ and $A \in R_j - X$.

Example 6: For the first database scheme of Example 5, we have $F_1 = \{A \to B, A \to C\}$, $F_2 = \{B \to D\}$, and $F_3 = \{C \to D\}$. ABC does not satisfy the uniqueness condition, since $CD \subseteq (ABC)^+_{F-F_3}$ where C is a key of CD and D is another attribute of CD. In the second database scheme of Example 5, $F_1 = \{A \to B, A \to C\}$, $F_2 = \{B \to D\}$, and $F_3 = \{C \to E\}$. The relation scheme ABC satisfies the uniqueness condition. In proof, $(ABC)^+_{F-F_3}$ = ABCD and only C appears also in CE and it is a key of CE; $(ABC)^+_{F-F_2}$ = ABCE and only B appears also in BD and it is a key of BD. []

```
    begin
(1) Y := X;
(2) while there is a key $V \varepsilon K_j$ such that $V \subseteq Y$ and $R_j \not\subseteq Y$ do
(3)     Y := $YR_j$
    end
```

Figure 1

We now show that the uniqueness condition implies that R_i^+ is uniquely derived. Suppose that R_i satisfies the uniqueness condition, and let $R_j \subseteq R_i^+$ $(j \neq i)$. Let $X = (R_i)^+_{F-F_j} \cap R_j$. We claim that X is a key of R_j. In proof, if X does not contain a key of R_j, then it is impossible to have $R_j \subseteq R_i^+$; and if it properly contains a key, then the uniqueness condition is violated. If $B \varepsilon X$, we say that R_j uses B in R_i^+ (sometimes we say that R_j uses X in R_i^+). If $B \varepsilon R_j - X$, we say that R_j adds B to R_i^+. We also say that R_i adds all its attributes to R_i^+ and uses none of them. The following propositions show that if R_i satisfies the uniqueness condition, then R_i^+ is uniquely derived in the sense that each $A \varepsilon R_i^+$ is added by a unique R_j.

Proposition 1: If R_i satisfies the uniqueness condition and $R_j \subseteq R_i^+$, then every $A \varepsilon R_j$ is either used by R_j in R_i^+ or added by R_j to R_i^+ (but not both).

Proof: Immediate from the definitions. []

<u>Proposition 2</u>: If R_i satisfies the uniqueness condition, then each $A \in R_i^+$ is added by a unique $R_j \subseteq R_i^+$.

<u>Proof</u>: Clearly, each $A \in R_i^+$ is added by at least one R_j. Suppose that for some $A \in R_i^+$ there are distinct R_q and R_p such that both R_q and R_p add A to R_i^+. Since R_q adds A, there is a key $X \in K_q$ such that $X \subseteq (R_i)^+_{F-F_q}$ and $A \in R_q - X$. Similarly, there is a key $Y \in K_p$ such that $Y \subseteq (R_i)^+_{F-F_p}$ and $A \in R_p - Y$. Consider a computation of R_i^+ by the algorithm of Fig. 1.

<u>Case 1</u>. $q \neq i$ and R_q is never used in line 3 of the algorithm. Thus, $XA \subseteq R_q \subseteq (R_i)^+_{F-F_q}$ and, so, R_i does not satisfy the uniqueness condition (a contradiction).

<u>Case 2</u>. $p \neq i$ and R_p is never used in line 3. This is similar to Case 1.

<u>Case 3</u>. R_q is used in line 3 after R_p (this includes the case where $p = i$, i.e., R_p is not used at all in line 3). Let Y' be the value of Y just before R_q is used in line 3. Since $Y' \subseteq (R_i)^+_{F-F_q}$ and R_p has already been used (i.e., $R_p \subseteq Y'$), it follows that $R_p \subseteq (R_i)^+_{F-F_q}$. But $A \in R_p$ and, so, $A \in (R_i)^+_{F-F_q}$. Since $X \subseteq (R_i)^+_{F-F_q}$, $X \in K_q$ and $A \in R_q - X$, the uniqueness condition is violated by R_i (a contradiction).

<u>Case 4</u>. R_p is used in line 3 after R_q. This is similar to Case 3.

Since at least one of q and p is different from i, no other case is possible. []

4. Main Theorem

Let $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ be a database scheme, and suppose that each relation r_i satisfies F_i and does not have any null values. In this section we will prove that the representative instance satisfies F if and only if each R_i satisfies the uniqueness condition. First, we will prove two lemmas that are needed for the "if" part (which is more difficult to prove). So, we assume that each R_i satisfies the uniqueness condition. Let $r = \biguplus_{i=1}^{n} \alpha_U(r_i)$ and consider the computation of $\text{chase}_F(r)$. Suppose that μ is a tuple of r that has originated from r_i. We say that R_j adds A to μ if R_j adds A to R_i^+. Similarly, R_j uses A in μ if R_j uses A in R_i^+. By the uniqueness condition and Propositions 1 and 2, for each $A \in R_i^+$, there are unique R_j and key $X \in K_j$ such that R_j adds A to μ using X (clealry, $A \in R_j - X$).

Lemma 1: Suppose that all the R_i´s satisfy the uniqueness condition. Let $R_p \subseteq R_i^+$, $Y \in K_p$, and $A \in R_p - Y$. If R_p either does not add A to R_i^+ or does not use Y in R_i^+, then there is $B \in Y$ such that R_p adds B to R_i^+.

Proof: Suppose that R_p does not add A to R_i^+. By Proposition 1, there is a key $V \in K_p$ used by R_p in R_i^+ such that $A \in V$. If $Y \subseteq V$, then $YA \subseteq V$. Since $A \notin Y$, it follows that the key V properly contains the key Y of R_p. But this is impossible and, so, $Y - V \neq \phi$ and every $B \in Y - V$ is added by R_p to R_i^+.

Now suppose that R_p does not use Y. If R_p does not use any key (i.e., p=i), then R_p adds every $B \in Y$. If R_p uses another key $V \in K_p$, then $Y \not\subseteq V$ (otherwise, one key properly contains another key). Therefore, there is $B \in Y - V$ that is added by R_p. []

We assume that $\text{chase}_F(r)$ is computed by repeatedly applying the following rule to tuples of r.

<u>FD-rule</u>. If tuples μ_1 and μ_2 agree on some key $X \in K_i$ but disagree on some $B \in R_i - X$, then for all $A \in R_i - X$:

(1) If $\mu_1(A) \neq \mu_2(A)$ and $\mu_1(A)$ is null, then replace all occurrences of $\mu_1(A)$ in r with $\mu_2(A)$.

(2) If $\mu_1(A) \neq \mu_2(A)$ and $\mu_2(A)$ is null, then replace all occurrences of $\mu_2(A)$ in r with $\mu_1(A)$.

The above FD-rule is sufficient for computing $\text{chase}_F(r)$, since a cover of F is embodied in the database scheme.

<u>Lemma 2</u>: Suppose that all the R_i's satisfy the uniqueness condition. Let r' be an intermediate relation in the computation of $\text{chase}_F(r)$, and let $\mu_1, \mu_2 \in r'$.

(A) If $\mu_1(A) = \mu_2(A) = \delta_i$ for some column A, then

(A1) the same R_j adds A to both μ_1 and μ_2, and the same key $X \in K_j$ is used by R_j in both μ_1 and μ_2, and

(A2) $\mu_1[R_j] = \mu_2[R_j]$.

(B) If $\mu_1(A)$ is non-null, then $\mu_1[R_j] \in r_j$, where R_j is the unique relation scheme that adds A to μ_1.

<u>Proof</u>: Induction on the number of applications of the FD-rule that are used to obtain r' from r.

Basis. Zero applications. That is, $r' = r$ and, so, all null values in r' are distinct. Thus, part A is vacuously true. As for part B, let μ_1 be a tuple of r' that has originated from r_i. If $\mu_1(A)$ is non-null, then $A \in R_i$ and R_i adds A to μ_1. But $\mu_1[R_i] \in r_i$ and, so, part B is true for r'.

Induction. Suppose that r'' is obtained from r by $n-1 > 0$ applications, and r' is obtained from r'' by a single application. In particular, suppose that r' is obtained from r'' by equating tuples ν_1 and ν_2 using key Y of R_p (i.e., the FD-rule is applied to ν_1 and ν_2 that agree on Y in r''). By the inductive hypothesis, the lemma is true for r'', and we have to show that it is true also for r'.

Part A. Let μ_1 and μ_2 be tuples of r' such that $\mu_1(A) = \mu_2(A) = \delta_i$ for some column A. If $\mu_1(A) = \mu_2(A)$ also in r'', then by the inductive hypothesis, conditions (A1)-(A2) of the lemma are satisfied in r'' and, hence, also in r' (if for some column B, we have $\mu_1(B) = \mu_2(B)$ in r'', then $\mu_1(B) = \mu_2(B)$ also in r'). Thus, we have to consider only μ_1, μ_2 and an A such that

(1) $\mu_1(A) = \nu_1(A)$ in r'',

(2) $\mu_2(A) = \nu_2(A)$ in r'', and

(3) $\nu_1(A) \neq \nu_2(A)$ in r'' and $A \in R_p - Y$.

(Otherwise, either $\mu_1(A) = \mu_2(A)$ in r'' or $\mu_1(A) \neq \mu_2(A)$ in r'.)

Case 1. R_p adds A using Y to both ν_1 and ν_2. Since $\nu_1(A) = \mu_1(A)$ in r'' (and $\mu_1(A)$ is null in r'', since it is null in r'), the inductive hypothesis implies that R_p adds A using Y also to μ_1, and

(i) $\mu_1[R_p] = \nu_1[R_p]$ in r".

Similarly, R_p adds A using Y also to μ_2 and

(ii) $\mu_2[R_p] = \nu_2[R_p]$ in r".

It remains to be shown that $\mu_1[R_p] = \mu_2[R_p]$ in r′. Suppose that $\nu_1(B)$ is non-null in r" for some B ε R_p - Y. Since R_p adds B to ν_1, part B of the inductive hypothesis implies that $\nu_1[R_p]$ ε r_p in r". By (i), $\mu_1[R_p]$ ε r_p in r" and, so, $\mu_1(A)$ is non-null in r′, contrary to assumption. Therefore, $\nu_1(B)$ is null in r" for all B ε R_p - Y, and similarly for ν_2. Since ν_1 and ν_2 are equated using key Y of R_p, it follows that $\nu_1[R_p] = \nu_2[R_p]$ in r′. By (i) and (ii), $\mu_1[R_p] = \mu_2[R_p]$ in r′.

<u>Case 2</u>. R_p does not add A to ν_1. We will show that this case cannot actually occur. Let r_k be the relation from which ν_1 has originated. By a lemma of [BDB], $R_p \subseteq R_k^+$ (since ν_1 is equated with ν_2 using R_p). Lemma 1 implies that there is B ε Y that is added by R_p to ν_1, because A is not added by R_p and A ε R_p - Y (by (3)). Since ν_1 and ν_2 are equated using key Y of R_p, it follows that $\nu_1(B) = \nu_2(B)$ in r". If $\nu_1(B)$ is null in R", then by part A of the inductive hypothesis, $\nu_1(A) = \nu_2(A)$ in r", contrary to (3). If $\nu_1(B)$ is non-null in r", then by part B of the inductive hypothesis, $\nu_1[R_p]$ ε r_p in r" and, by (1), $\mu_1(A)$ is non-null in r′, contrary to assumption. Thus, $\nu_1(B)$ is neither null nor non-null in r" and, consequently, this case cannot occur.

<u>Case 3</u>. R_p does not add A to ν_2. This is similar to Case 2.

<u>Part B</u>. We have to show that if $\mu_1(A)$ is non-null in r′, then $\mu_1[R_j]$ ε r_j, where R_j adds A to μ_1. If $\mu_1(A)$ is also non-null in r",

then by the inductive hypothesis, $\mu_1[R_j] \varepsilon r_j$ in r" and, so, $\mu_1[R_j]$ is the same in r" and r´ (since all the columns of $\mu_1[R_j]$ in r" are non-null and cannot be changed). Thus, we have to consider only the following case.

(a) $\nu_1(A)$ is non-null in r",

(b) $\nu_2(A)$ is null in r",

(c) $\mu_1(A) = \nu_2(A)$ in r", and

(d) ν_1 and ν_2 are equated using a key Y of R_p and A ε R_p - Y.

<u>Claim 1</u>: $\nu_1[R_p] \varepsilon r_p$ in r" (and, hence, also in r´).

<u>Proof</u>: <u>Case 1</u>. R_p adds A to ν_1. Since $\nu_1(A)$ is non-null in r" (by (a)), the inductive hypothesis implies that $\nu_1[R_p] \varepsilon r_p$ in r". Therefore, $\nu_1[R_p]$ is non-null in r" and, hence, it is the same in r" and r´.

<u>Case 2</u>. R_p does not add A to ν_1. By (d) and Lemma 1, there is B ε Y that is added by R_p to ν_1. Suppose that $\nu_1(B)$ is null. Since $\nu_1(B) = \nu_2(B)$ in r" (by (d)), part A of the inductive hypothesis implies that $\nu_1(A) = \nu_2(A)$ in r". This is contrary to (a) and (b) and, therefore, $\nu_1(B)$ is non-null in r". By part B of the inductive hypothesis, $\nu_1[R_p] \varepsilon r_p$ in r". []

<u>Claim 2</u>: R_p uses Y in ν_2.

<u>Proof</u>: Suppose that R_p does not use Y in ν_2. By Lemma 1, there is B ε Y such that R_p adds B to ν_2. By (d), $\nu_2(B) = \nu_1(B)$ in r" and, by Claim 1, $\nu_2(B)$ is non-null in r". By the inductive hypothesis, $\nu_2[R_p] \varepsilon r_p$ in r". But this is impossible, since $\nu_2(A)$ is null in

r". []

By Claim 2 and (d), R_p adds A to ν_2 using Y. Therefore, part A of the inductive hypothesis, (b) and (c) imply that $\mu_1[R_p] = \nu_2[R_p]$ in r" (and R_p adds A to μ_1 using Y). If $\nu_2(C)$ is non-null in r" for some $C \varepsilon R_p - Y$, then by the inductive hypothesis, $\nu_2(A)$ is non-null, contrary to (b). Hence, $\nu_2(C)$ is null in r" for all $C \varepsilon R_p - Y$ and, by (d), $\nu_1[R_p] = \nu_2[R_p]$ in r′. Thus, $\mu_1[R_p] = \nu_2[R_p] = \nu_1[R_p] \varepsilon r_p$ in r′. []

Corollary 1: Suppose that during the computation of $\text{chase}_F(r)$, the null in column A of μ_1 is replaced with a non-null as a result of equating ν_1 and ν_2 using key Y of R_p. Then R_p adds A to μ_1 using Y.

Proof: Let ν_1, ν_2, and μ_1 be the same as in part B of the proof of Lemma 3. We have shown that R_p adds A to μ_1 using Y. []

Theorem 1: Let $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ be a database scheme. Suppose that each relation r_i satisfies F_i and does not have any nulls. Then the representative instance satisfies F if and only if each R_i satisfies the uniqueness condition.

Proof: If. Suppose that some FD $X \rightarrow A \varepsilon F_i$ is violated in the representative instance. That is, there are tuples μ_1 and μ_2 in the representative instance such that

(1) $\mu_1[X] = \mu_2[X]$, and

(2) $\mu_1(A)$ and $\mu_2(A)$ are distinct non-nulls.

<u>Claim 1</u>: $\mu_1[R_i] \in r_i$ and $\mu_2[R_i] \in r_i$.

<u>Proof</u>: We will prove that $\mu_1[R_i] \in r_i$ (proving that $\mu_2[R_i] \in r_i$ is similar).

<u>Case 1</u>. R_i adds A to μ_1. By Lemma 2, $\mu_1[R_i] \in r_i$.

<u>Case 2</u>. R_i does not add A to μ_1. By Lemma 1, there is $B \in X$ such that B is added by R_i to μ_1. Suppose that $\mu_1(B)$ is null. By Lemma 2, $\mu_1(A) = \mu_2(A)$, since $\mu_1(B) = \mu_2(B)$. This contradiction implies that $\mu_1(B)$ is non-null and, by Lemma 2, $\mu_1[R_i] \in r_i$. []

By Claim 1 and (1), $\mu_1(A) = \mu_2(A)$, since r_i satisfies F_i. But this is contrary to (2) and, so, the "if" part is proved.

<u>Only If</u>. Suppose that some R_i does not satisfy the uniqueness condition. That is, there is an R_j ($j \neq i$), a key $X \in K_j$, and an attribute $A \in R_j - X$ such that $XA \subseteq (R_i)^+_{F-F_j}$. We have to show that there are relations $r_1,\ldots,r_n$ (without nulls) that satisfy $F_1,\ldots,F_n$, respectively, such that the representative instance does not satisfy F. We choose $r_1,\ldots,r_n$ as follows. For $k \neq j$, each r_k has exactly one tuple that maps all the attributes of R_k to 1. The relation r_j has exactly one tuple that maps A to 2 and each attribute in $R_j - A$ to 1. Let $r = \bigcup_{k=1}^{n} \alpha_U(r_k)$, and let μ_k be the tuple of r that has originated from r_k. We apply the chase process to r using only the FDs of $F - F_j$ and tuples μ_k such that $k \neq j$. Let r′ be the resulting relation. By a Lemma of [BDB], $\mu_i(B) = 1$ for all $B \in (R_i)^+_{F-F_j}$ in r′. Therefore, tuples μ_i and μ_j of r′ violate $X \rightarrow A$. Clearly, if we have a violation of $X \rightarrow A$ in r′,

then this violation exists also in the representative instance and, so, the proof is complete. []

5. Computing Total Projections of the Representative Instance

Suppose that $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ is a database scheme such that each R_i satisfies the uniqueness condition, and let X be a set of attributes. In this section we will show how to construct in polynomial time an expression E whose value is the X-total projection of the representative instance. The expression E is of the form $\biguplus_j \pi_X(E_j)$, where each E_j is an extension join. Among all the expressions of this form whose value is the X-total projection of the representative instance, the expression E is minimal in both the number of extension joins and the number of join operators.

Let $r = \biguplus_{i=1}^{n} \alpha_U(r_i)$, and $\mu \in chase_F(r)$ be a tuple that has originated from r_i. We define

$$P = \{A \mid \mu(A) \text{ is non-null}\}$$

Consider the set

$$S = \{R_j \mid \text{there is an } A \in P \text{ such that } R_j \text{ adds } A \text{ to } \mu\}$$

By Lemma 2, $P = R_{j_1}\ldots R_{j_m}$, where $S = \{R_{j_1},\ldots,R_{j_m}\}$. Let R_{j_p} and R_{j_q} be distinct members of S. By Corollary 1 and Lemma 2, the columns of μ for the attributes added by R_{j_p} become non-null as a result of a single application of R_{j_p}, and similarly for R_{j_q}. Therefore, the columns of both R_{j_p} and R_{j_q} cannot become non-null simultaneously (because R_{j_p} and R_{j_q} add distinct attributes). Consequently, let $R_{j_1},R_{j_2},\ldots,R_{j_m}$ be an ordering of the elements of S such that if $p < q$, then $\mu[R_{j_p}]$ becomes non-null before $\mu[R_{j_q}]$. Note that $j_1 = i$.

Lemma 3: $\bowtie_{k=1}^{m} R_{j_k}$ is an extension join of R_i, and $\mu[P] \varepsilon \bowtie_{k=1}^{m} R_{j_k}$.

Proof: Consider μ just before $\mu[R_{j_p}]$ (for some $1 < p < m$) becomes non-null. Let $\mu(B)$ be non-null, and suppose that R_{j_q} adds B to R_i^+. By Lemma 2, $\mu[R_{j_q}]$ is non-null and, so, $q < p$. Therefore, the non-null columns of μ are $R_{j_1} \ldots R_{j_{p-1}}$. By Corollary 1, $\mu[R_{j_p}]$ becomes non-null as a result of applying R_{j_p} and, so, $Y \underline{C} R_{j_1} \ldots R_{j_{p-1}}$ for some key Y of R_{j_p}. Thus, $\bowtie_{k=1}^{m} R_{j_k}$ is an extension join (of R_i).

Now consider μ at the end of the chase process. By Lemma 2, $\mu[R_{j_k}] \varepsilon r_{j_k}$ for every $1 < k < m$, and μ is non-null exactly in the columns of $P = R_{j_1} \ldots R_{j_m}$. Therefore, $\mu[P] \varepsilon \bowtie_{k=1}^{m} R_{j_k}$. []

Lemma 3 states that the non-null portion of any tuple in the representative instance is in some extension join. Conversely, if $\bowtie_{k=1}^{m} R_{j_k}$ is an extension join and $\mu \varepsilon \bowtie_{k=1}^{m} R_{j_k}$, then by a lemma of [BDB], there is a tuple ν in the representative instance such that $\nu[P] = \mu$. Therefore, we have the following corollary.

Corollary 2: If all the R_i's satisfy the uniqueness condition, then the X-total projection of the representative instance is given by the expression $\biguplus_j \pi_X(E_j)$, where each E_j is an extension join over a set of attributes containing X.

The expression E of Corollary 2 might have redundant subexpressions. We now show how to minimize it. Clearly, there is an extension join of R_{j_1} over a set of attributes containing X only if $X \subseteq R_{j_1}^+$. Furthermore, if $\bowtie_{k=1}^{p} R_{j_k}$ $(p < m)$ is also an extension join of R_{j_1} over a set of attributes containing X, then $\pi_X(\bowtie_{k=1}^{m} R_{j_k}) \subseteq \pi_X(\bowtie_{k=1}^{p} R_{j_k})$. Therefore, we need to consider only <u>minimal</u> extension joins of R_{j_1} with respect to X, that is, extension joins $\bowtie_{k=1}^{m} R_{j_k}$ such that

(1) $X \subseteq R_{j_1} \ldots R_{j_m}$, and

(2) no proper subsequence of $R_{j_1}, \ldots, R_{j_m}$ is an extension join of R_{i_1} that satisfies (1).

<u>Lemma 4</u>: If $X \subseteq R_i^+$, and $R_1, \ldots, R_n$ satisfy the uniqueness condition, then the minimal extension join of R_i with respect to X is unique and can be found in linear time.

<u>Proof</u>: The algorithm of Fig. 2 computes a set S of relation schemes that must be included in any extension join of R_i over a set of attributes containing X. The idea is that S must contain any R_j that adds an attribute $A \in X$ to R_i^+ and, recursively, if $R_k \in S$, then S includes also every R_j that adds an attribute of R_k. Now consider an ordering $R_{j_1}, \ldots, R_{j_p}$ of the elements of S such that if $k < m$, then R_k is used before R_m in line 3 of Fig. 1 during the computation of R_i^+. (Note that $j_1 = i$.)

<u>Claim 1</u>: $\bowtie_{k=1}^{p} R_{j_k}$ is an extension join (of R_i).

<u>Proof</u>: Let $V = R_{j_1} \ldots R_{j_m}$, and Y be the key used by $R_{j_{m+1}}$ in R_i^+ $(1 \leq m < p)$. Consider an attribute $B \in Y$, and suppose that R_q adds B to R_i^+. R_q must have been added to S in line 4 of Fig. 2, since $R_{j_{m+1}} \in S$. Further, R_q must be used before $R_{j_{m+1}}$ during the computation of R_i^+.

```
    begin
(1) S := $\phi$;
(2) Y := X;
(3) while there is an $R_j$ such that $R_j \notin S$ and $R_j$ adds some $A \in Y$ to $R_i^+$ do
        begin
(4)     S := S $\cup$ {$R_j$};
(5)     Y := Y$R_j$
        end
    end
```

Figure 2

Therefore, if $B \in Y$, then $B \in V$ and, so, $\bowtie_{k=1}^{p} R_{j_k}$ is an extension join. []

The algorithm of Fig. 2 can be implemented in linear time using a data structure similar to that of [BB]. Similarly, the ordering of the elements of S can be determined in linear time. []

We have shown that we need to consider only minimal extension joins of those R_j such that $X \subseteq R_j^+$. Now suppose that E_i and E_j are minimal extension joins of R_i and R_j, respectively, with respect to X. The following lemma implies that if E_j includes R_i, then $\pi_X(E_j) \subseteq \pi_X(E_i)$.

Lemma 5: If E_j includes R_i, then E_j includes every relation scheme that appears in E_i.

Proof: Apply the algorithm of Fig. 2 to X and R_i, and let $R_{i_1},\ldots,R_{i_p}$ be the order in which the elements of S are added in line 4. We will prove by induction on k that $XR_{i_1}\ldots R_{i_k} \subseteq V$ and R_{i_k} is in E_j, where V is the set of attributes in E_j.

Basis. k = 0. Obvious.

Induction. Suppose that R_{i_k} is not included in E_j (i.e., $i_k \neq i$). Since R_{i_k} is added after $R_{i_1},\ldots,R_{i_{k-1}}$ in line 4, R_{i_k} adds some $B \in XR_{i_1}\ldots R_{i_{k-1}}$ to R_i^+ using a key Y (i.e., $Y \subseteq (R_i)^+_{F-F_{i_k}}$). By the inductive hypothesis, $B \in V$. Since E_j is an extension join of R_j that does not include R_{i_k}, it follows that $V \subseteq (R_j)^+_{F-F_{i_k}}$. Moreover, $Y \subseteq (R_j)^+_{F-F_{i_k}}$, because $Y \subseteq (R_i)^+_{F-F_{i_k}}$ and R_i (but not R_{i_k}) is included in E_j. Thus, $YB \subseteq (R_j)^+_{F-F_{i_k}}$, and so R_j violates the uniqueness condition. []

Theorem 2: Let $\langle R_1,K_1\rangle,\ldots,\langle R_n,K_n\rangle$ be a database scheme such that each R_i satisfies the uniqueness condition. Suppose that each relation r_i satisfies F_i and does not have any nulls. An expression whose value is the X-total projection of the representative instance can be found in $O(n^2)$ time, where n is the space needed to write down the database scheme.

Proof: The algorithm for constructing the expression is given in Fig. 3. For a given value of T, define

$$E(T) = \{E_k \mid E_k \text{ is the minimal extension join of } R_k \text{ with respect to } X, \text{ and } R_k \in T\}$$

Let F(T) be the union of all extension joins in E(T) projected onto X.

```
    begin
(1) W := $\phi$;
(2) Let T = $\{R_i \mid X \subseteq R_i^+\}$;
(3) for every $R_i \in T$ do
        begin
(4)     let $E_i$ be the minimal extension join of $R_i$ with respect to X;
(5)     if there is no $R_j \in T$ $(j \neq i)$ that appears in $E_i$ then
(6)        add $E_i$ to W
(7)     else remove $R_i$ from T
        end;
(8) let W = $\{E_1,\dots,E_m\}$;
(9) return the expression $\bigcup_{j=1}^{m} \pi_X(E_j)$
    end
```

Figure 3

Using Lemma 5, we can easily prove by induction that for all values of T obtained in line 3, $F(T) \equiv F(T_0)$, where T_0 is the initial value of T. By Corollary 2 and Lemma 4, the value of $F(T_0)$ is equal to the X-total projection of the representative instance. The expression returned by the algorithm of Fig. 3 is $F(T_f)$, where T_f is the final value of T. Thus, the algorithm returns an expression whose value is the X-total projection of the representative instance.

By Lemma 4 and [BB], the algorithm of Fig. 9 can be implemented to run in $O(n^2)$ time. []

Corollary 3: Among all the expressions of the form $\bigcup_j \pi_X(E_j)$, whose value is the X-total projection of the representative instance, the expression returned by the algorithm of Fig. 3 is minimal in both the number of subexpressions of the form $\pi_X(E_j)$ and the total number of join operators.

Proof: The expression returned in line 9 is nonredundant in the sense that if any subexpression of the form $\pi_X(E_j)$ is removed from the union, then we can find relations $r_1,\ldots,r_n$ for which the value of the resulting expression is not the X-total projection of the representative instance. In proof, for each R_i that appears in E_j, the relation r_i has exactly one tuple with 1 in every column; all the other relations are empty. Clearly, the value of E_j is a relation with exactly one tuple that has 1 in every column. All the other extension joins have at least one empty relation and, so, their value is the empty relation. Thus, removing $\pi_X(E_j)$ changes the value of the expression from a relation with one tuple to the empty relation. By a theorem of [SY], it follows that the expression returned in line 9 is minimal in the number of subexpressions of the form $\pi_X(E_j)$. Since each E_j has no equivalent expression with fewer join operators (by Lemma 4), the expression returned in line 9 is also minimal in the total number of join operators (by a result of [SY]). []

Example 7: Suppose that the attributes are P(project), D(department), M(manager), L(location), and A(assistant), and the database scheme is <LDP,{LD}>,<DPM,{DP}>,<LMA,{LM}>.

Intuitively, the database scheme describes an application in which each project belongs to several departments and is carried out in several locations, but a department can have only one project in each location. In each department participating in a project, there is a manager responsible for that project. Each manager has an assistant in each location. These relation schemes satisfy the uniqueness condition.

Suppose we want to compute the total projection of the representative instance onto LM. After Step (1) of the algorithm, S = {LDP,LMA}. The minimal extension join of LDP with respect to LM is LDP $\bowtie$ PDM, and the minimal extension join of LMA with respect to LM is LMA. In Step (2) of the algorithm S is not changed, and so an expression for the LM-total projection of the representative instance is $\pi_{LM}(LDP \bowtie PDM) \uplus \pi_{LM}(LMA)$.

The result of the above expression is all tuples (l,m) such that either manager m has an assistant in location l or manager m manages some project in location l. Considering the fact that there might be partial information (e.g., a manager with a project in a location where he does not have an assistant, or a manager with an assistant in a location where he does not have a project), the correct answer is indeed given by the above expression. []

6. Conclusions

We have proposed the representative instance as a measure of the data stored in the database, and we have characterized, in terms of the uniqueness condition, the database schemes for which the representative instance always satisfies the functional dependencies. The uniqueness condition can be viewed as an extension of Boyce-Codd normal form, since it removes inter-relation anomalies in the sense that each relation can be updated independently of the contents of the other relations without violating global consistency. (It is an extension of Boyce-Codd normal form, since a relation scheme that satisfies the uniqueness condition is also in Boyce-Codd normal form.) This condition is much less restrictive than [BG], and we believe that many practical applications satisfy it. We have also shown how to compute efficiently total projections of the representative instance if the uniqueness condition is satisfied.

In [Sa] we have dealt with database schemes that do not satisfy the uniqueness condition. We have shown that the representative instance can be guaranteed to satisfy the functional dependencies if the modified foreign-key constraint is imposed on the database. The modified foreign-key constraint is less restrictive and semantically more meaningful than imposing the existence of a universal instance. Under this constraint, total projections of the representative instance can be computed by performing the union of several extension joins.

References

[ABU] Aho, A. V., C. Beeri, and J. D. Ullman, "The Theory of Joins in Relational Databases," ACM Trans. on Database Systems, Vol. 4, No. 3 (Sept. 1979), pp. 297-314.

[ASU] Aho, A. V., Y. Sagiv, and J. D. Ullman, "Equivalences Among Relational Expressions," SIAM J. Computing, Vol. 8, No. 2 (May 1979), pp. 218-246.

[Arm] Armstrong, W. W., "Dependency Structures of Database Relationships," Proc. IFIP 74, North Holland, 1974, pp. 580-583.

[BB] Beeri, C. and P. A. Bernstein, "Computational Problems Related to the Design of Normal Form Relational Schemas," ACM Trans. on Database Systems, Vol. 4, No. 1 (March 1979), pp. 30-59.

[Ber] Bernstein, P. A., "Synthesizing Third Normal Form Relations from Functional Dependencies," ACM Trans. on Database Systems, Vol. 1, No. 4 (Dec. 1976), pp. 277-298.

[BG] Bernstein, P. A., and N. Goodman, "What Does Boyce-Codd Normal Form Do?," Proc. Int. Conf. on Very Large Data Bases, 1980, pp. 245-259.

[BDB] Biskup, J., U. Dayal, and P. A. Bernstein, "Synthesizing Independent Database Schemas," Proc. ACM-SIGMOD Int. Conf. on Management of Data, 1979, pp. 143-151.

[C1] Codd, E. F., "A Relational Model for Large Shared Data Banks," Comm. ACM, Vol. 13, No. 6 (June 1970), pp. 377-387.

[C2] Codd, E. F., "Extending the Database Relational Model to Capture More Meaning," ACM Trans. on Database Systems, Vol. 4, No. 4 (Dec. 1979), pp. 397-434.

[FMU] Fagin, R., A. O. Mendelzon, and J. D. Ullman, "A Simplified Universal Relation Assumption and Its Properties," IBM research report, RJ2900, Nov., 1980.

[Ho1] Honeyman, P., "Testing Satisfaction of Functional Dependencies," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[Ho2] Honeyman, P., "Extension Joins," Proc. Int. Conf. on Very Large Data Bases, 1980, pp. 239-244.

[HLY] Honeyman, P., R. E. Ladner, and M. Yannakakis, "Testing the Universal Instance Assumption," Information Processing Letters, Vol. 10, No. 1 (Feb. 1980), pp. 14-19.

[Ko] Korth, H. F., "A Proposal for the SYSTEM/U Query Language," unpublished memorandum, Stanford University, Stanford, CA, 1980.

[KU] Korth, H. F. and J. D. Ullman, "System/U: a Database System Based on the Universal Relation Assumption," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[Li] Lien, Y. E., "Multivalued Dependencies With Null Values in Relational Databases," Proc. Int. Conf. on Very Large Data Bases, 1979.

[Ma] Maier, D., "Discarding the Universal Instance Assumption: Preliminary Results," Proc. XP1 Conf., Stonybrook, N. Y., June, 1980.

[MMS] Maier D., A. O. Mendelzon, and Y. Sagiv, "Testing Implications of Data Dependencies," ACM Trans. on Database Systems, Vol. 4, No. 4 (Dec. 1979), pp. 455-469.

[MU] Maier, D., and J. D. Ullman, "Maximal Objects and the Semantics of

Universal Relation Databases," Tech. Rept. 80-016, Dept. of Computer Science, State University of New York at Stony Brook, Stony Brook, NY, Nov. 1980.

[Sa] Sagiv, Y., "Can We Use the Universal Instance Assumption Without Using Nulls?" Proc. ACM-SIGMOD Int. Conf. on Management of Data, Ann Arbor, April 1981, pp. 108-120.

[SY] Sagiv, Y., and M. Yannakakis, "Equivalences Among Relational Expressions with the Union and Difference Operators," J. ACM, Vol. 27, No. 4 (Oct. 1980), pp. 633-655.

[Sc] Sciore, E., "The Universal Instance and Database Design," TR 271, Dept. of Elec. Eng. and Comp. Sci., Princeton University, Princeton, N. J., June, 1980.

[Va1] Vassiliou, Y., "Functional Dependencies and Incomplete Information," Proc. Int. Conf. on Very Large Data Bases, 1980, pp. 260-269.

[Va2] Vassiliou, Y., "A Formal Treatment of Imperfect Information in Database Management," Technical Report CSRG-123, University of Toronto, Nov., 1980.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-81-1052	2.	3. Recipient's Accession No.
4. Title and Subtitle A Characterization of Globally Consistent Databases and their Correct Access Paths			5. Report Date July 1981
			6.
7. Author(s) Yehoshua Sagiv			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois 1304 W. Springfield Urbana, IL 61801			10. Project/Task/Work Unit No.
			11. Contract/Grant No. MCS-80-03308
12. Sponsoring Organization Name and Address National Science Foundation Washington, D.C.			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts We propose the representative instance as a representation of the data stored in a database whose relations are not the projections of a universal instance. We characterize the database schemes for which local consistency implies global consistency. (Local consistency means that each relation satisfies its own functional dependencies, and global consistency means that the representative instance satisfies all the functional dependencies.) We show how to compute efficiently projections of the representative instance provided that local consistency implies global consistency. Throughout this work we assume that a cover of the functional dependencies is embodied in the database scheme in the form of keys.

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 39
	20. Security Class (This Page UNCLASSIFIED	22. Price

FORM NTIS-35 (10-70) USCOMM-DC 40329-P71

UNIVERSITY OF ILLINOIS-URBANA
3 0112 084230058